Student : Esteban Ordenes

Post Graduate Program in Data Science and Business Analytics

PGP-DSBA-UTA-Dec20-A

# AllLifeBank Customer Segmentation

## Context

AllLife Bank wants to focus on its credit card customer base in the next financial year. They have been advised by their marketing research team, that the penetration in the market can be improved. Based on this input, the Marketing team proposes to run personalized campaigns to target new customers as well as upsell to existing customers. Another insight from the market research was that the customers perceive the support services of the back poorly. Based on this, the Operations team wants to upgrade the service delivery model, to ensure that customer's queries are resolved faster.

## Objective and Key Questions

Identify different segments in the existing customer based on their spending patterns as well as past interaction with the bank.

- Perform EDA
- Apply Clustering Algorithms.
- How many clusters are formed ?
- Analyze differences between discovered clusters.
- How are these segments different from each other?
- Provide recommendations to the bank on how to better market to and service these customers.

## Data Dictionary

Data is of various customers of a bank with their credit limit, the total number of credit cards the customer has, and different channels through which customer has contacted the bank for any queries, different channels include visiting the bank, online and through a call center.

- Sl_No : Serial Number
- Customer Key : Customer unique Key
- Avg_Credit_Limit : Average Credit Limit
- Total_Credit_Cards : Number of Credit Cards they have
- Total_visits_bank : Number of times they visited the bank
- Total_visits_online : Number of times they visited the website online
- Total_calls_made : Number of times they made a call

### Loading libraries

In [340…

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()
sns.set()

from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage,cophenet
from sklearn.preprocessing import StandardScaler
from scipy.spatial.distance import pdist
from scipy.cluster.hierarchy import dendrogram, linkage,cophenet
from sklearn.metrics import silhouette_score

from yellowbrick.cluster import KElbowVisualizer, SilhouetteVisualizer

import warnings
warnings.filterwarnings('ignore')

pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", 200)
```

## Read the dataset

In [341…

```python
CCData = pd.read_excel("Credit Card Customer Data.xlsx")
```

In [342…

```python
# copying data to another variable to avoid any changes to original data
data = CCData.copy()
```

## View the first and last 5 rows of the dataset.

In [343…

```python
data.head()
```

Out[343…

| | Sl_No | Customer Key | Avg_Credit_Limit | Total_Credit_Cards | Total_visits_bank | Total_visits_online | Total_calls |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 87073 | 100000 | 2 | 1 | 1 | |
| **1** | 2 | 38414 | 50000 | 3 | 0 | 10 | |
| **2** | 3 | 17341 | 50000 | 7 | 1 | 3 | |
| **3** | 4 | 40496 | 30000 | 5 | 1 | 1 | |
| **4** | 5 | 47437 | 100000 | 6 | 0 | 12 | |

In [344…

```python
data.tail()
```

Out[344…

| | Sl_No | Customer Key | Avg_Credit_Limit | Total_Credit_Cards | Total_visits_bank | Total_visits_online | Total_c |
|---|---|---|---|---|---|---|---|
| **655** | 656 | 51108 | 99000 | 10 | 1 | 10 | |
| **656** | 657 | 60732 | 84000 | 10 | 1 | 13 | |
| **657** | 658 | 53834 | 145000 | 8 | 1 | 9 | |

| | Sl_No | Customer Key | Avg_Credit_Limit | Total_Credit_Cards | Total_visits_bank | Total_visits_online | Total_c |
|---|---|---|---|---|---|---|---|
| **658** | 659 | 80655 | 172000 | 10 | 1 | 15 | |
| **659** | 660 | 80150 | 167000 | 9 | 0 | 12 | |

## Understand the shape of the dataset.

In [345…
```
data.shape
```

Out[345…
```
(660, 7)
```

- The dataset has 660 rows and 7 columns

## Check data types and number of non-null values for each column.

In [346…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 660 entries, 0 to 659
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Sl_No                660 non-null    int64
 1   Customer Key         660 non-null    int64
 2   Avg_Credit_Limit     660 non-null    int64
 3   Total_Credit_Cards   660 non-null    int64
 4   Total_visits_bank    660 non-null    int64
 5   Total_visits_online  660 non-null    int64
 6   Total_calls_made     660 non-null    int64
dtypes: int64(7)
memory usage: 36.2 KB
```

- `SL_No` and `Customer Key` seem to be serial numbers and unique row identifiers. These do not provide statistical significance and we will remove from the dataset.
- We can see that there are total 7 columns and 660 number of rows in the dataset.
- All columns' data type are integer.
- There are NO null values in the columns. We can further confirm this using isna() method.

In [347…
```
data.isna().sum()
```

Out[347…
```
Sl_No                  0
Customer Key           0
Avg_Credit_Limit       0
Total_Credit_Cards     0
Total_visits_bank      0
Total_visits_online    0
Total_calls_made       0
dtype: int64
```

In [348…
```
# remove "SL_No" and "Customer Key"feature from the dataset.
data.drop(['Sl_No','Customer Key'],axis=1,inplace=True)
```

## Check for duplicate data.

In [349…

```
data.duplicated().sum()
```

Out[349…

```
11
```

- There are 11 duplicate observations. We will remove them from the data.

In [350…

```
data = data[(~data.duplicated())].copy()
```

In [ ]:

```

```

## Summary of the dataset

In [351…

```
# Summary of continuous columns
data.describe().T
```

Out[351…

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Avg_Credit_Limit** | 649.0 | 34878.274268 | 37813.736638 | 3000.0 | 11000.0 | 18000.0 | 49000.0 | 200000.0 |
| **Total_Credit_Cards** | 649.0 | 4.708783 | 2.173763 | 1.0 | 3.0 | 5.0 | 6.0 | 10.0 |
| **Total_visits_bank** | 649.0 | 2.397535 | 1.625148 | 0.0 | 1.0 | 2.0 | 4.0 | 5.0 |
| **Total_visits_online** | 649.0 | 2.624037 | 2.952888 | 0.0 | 1.0 | 2.0 | 4.0 | 15.0 |
| **Total_calls_made** | 649.0 | 3.590139 | 2.877911 | 0.0 | 1.0 | 3.0 | 5.0 | 10.0 |

- `Avg_Credit_Limit` mean and median is 34878.27 and 18000.0 respectively.
- `Total_Credit_Cards` mean and median 4.708 and 5.0 respectively.
- `Total_visits_bank` mean and median is 2.397 and 2.0 respectively.
- `Total_visits_online` mean and median is 2.624 and 2.0 respectively,
- `Total_calls_made` mean and median is 3.590 and 3.0 respectively.

In [ ]:

```

```

# EDA

## Univariate analysis

In [352…

```
def histogram_boxplot(feature , figsize=(15,10) , bins=None):
    """ Histogram and Boxplot combined
    feature: 1-d feature array
    figsize: size of figg.default (15,10)
    bins: number of bins.default None/auto
    """
    mean = feature.mean()
    median = feature.median()
    mode = feature.mode()

    f2, (ax_box2 , ax_hist2) = plt.subplots(nrows = 2, # num of rows of the subplot. gr
                                            sharex = True, # x-axis will be shared amon
                                            gridspec_kw = { "height_ratios": (.25 , .75
```

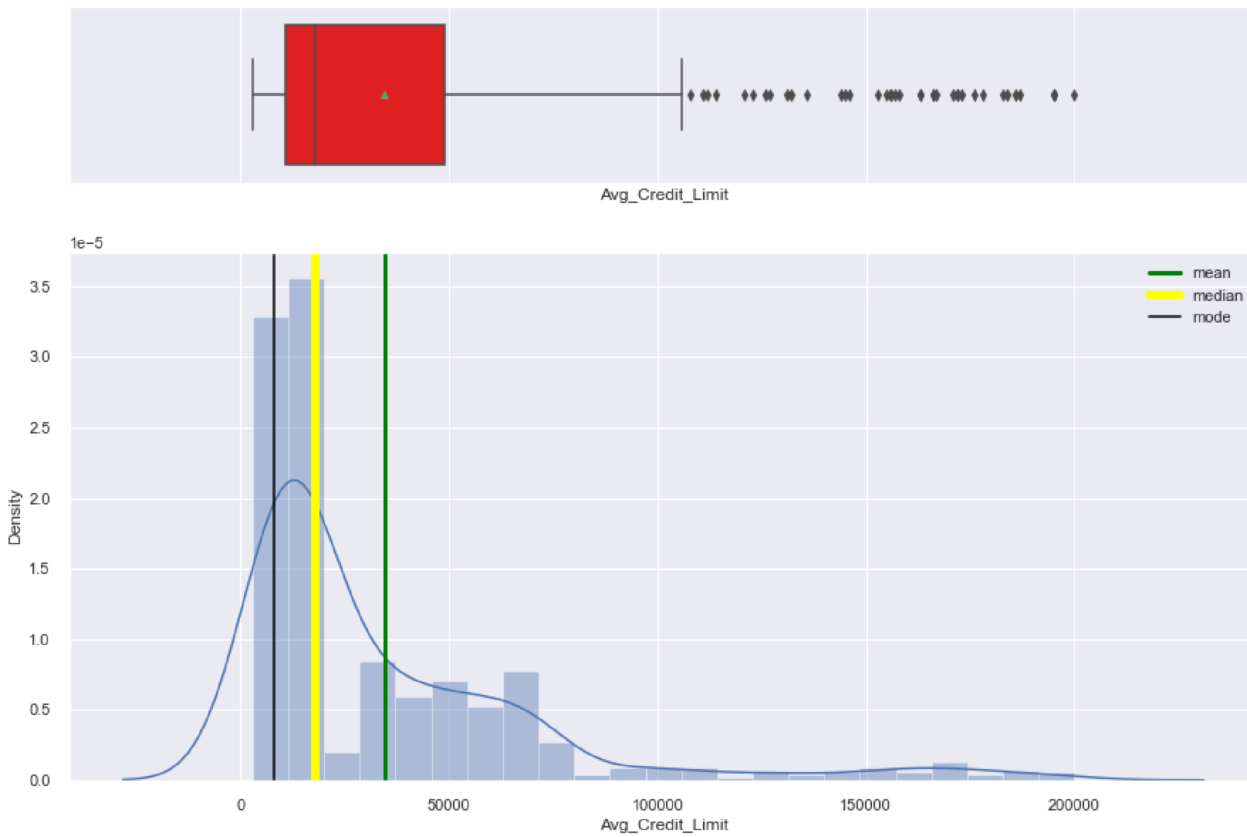
```
                                          figsize = figsize
                                         ) # create the 2 subplots

    sns.boxplot(feature , ax = ax_box2 , showmeans = True , color = 'red') # boxplot wi
    if bins:
        sns.distplot(feature , kde = True , ax = ax_hist2, bins = bins)
    else:
        sns.distplot( feature , kde = True , ax = ax_hist2 )
    ax_hist2.axvline( mean , color = 'green' , linestyle='-' , linewidth = 3 , label =
    ax_hist2.axvline( median , color = 'yellow' , linestyle='-' , linewidth = 6 , label
    ax_hist2.axvline( mode[0] , color = 'black' , linestyle='-' , label = 'mode' ) # ad
    ax_hist2.legend()

    print( 'Mean:'+ str( mean ) )
    print( 'Median:'+ str( median ) )
    print( 'Mode:'+ str( mode[0] ) )
```

## Observation on Avg_Credit_Limit

In [353...

```
histogram_boxplot(data.Avg_Credit_Limit)
```

```
Mean:34878.274268104775
Median:18000.0
Mode:8000
```



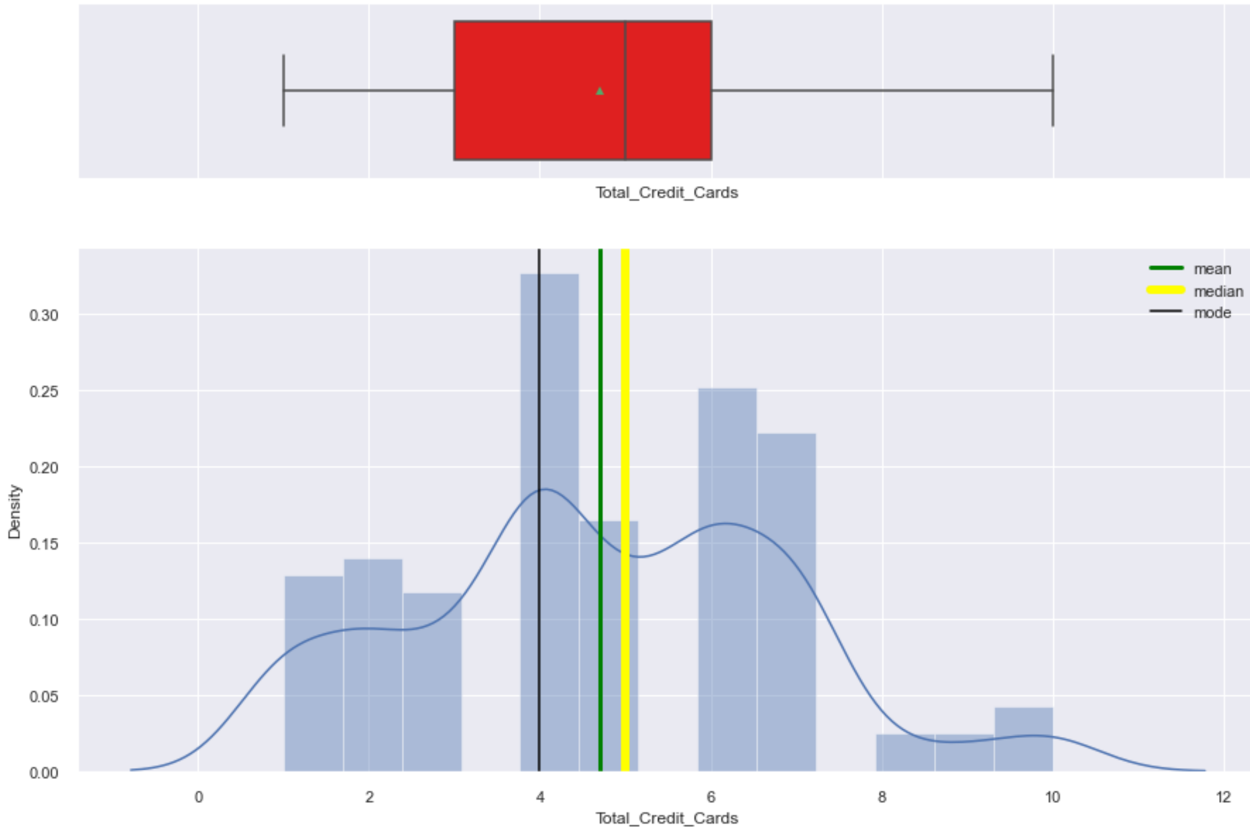

- Avg_Credit_Limit feature is right-skewed.
- There are many outliers to the right of the curve.

## Observation on Total_Credit_Cards

In [354...

```
histogram_boxplot(data.Total_Credit_Cards)
```

```
Mean:4.708782742681048
Median:5.0
Mode:4
```



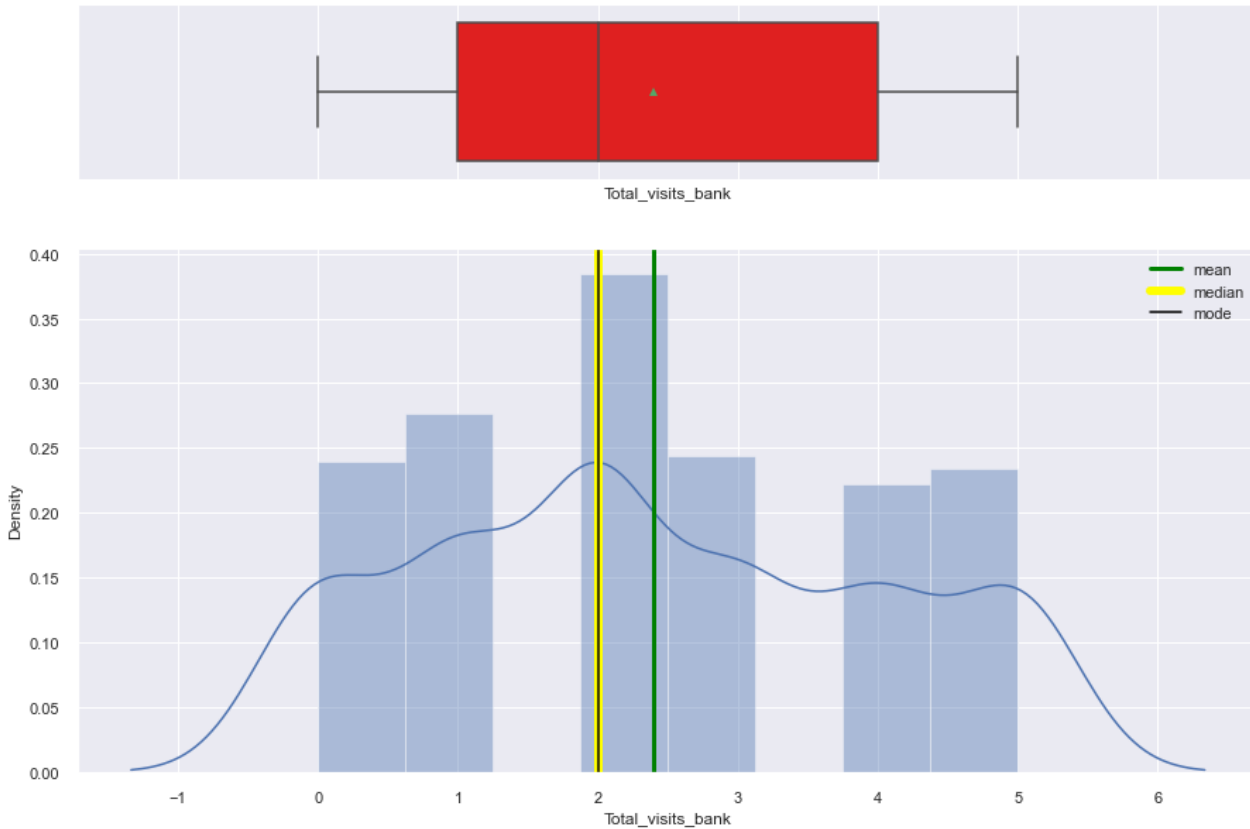

- Total_Credit_Cards feature seems fairly normal distributed.
- There are no outliers in the distribution.

## Observation on Total_visits_bank

In [355…

```
histogram_boxplot(data.Total_visits_bank   )
```

```
Mean:2.3975346687211094
Median:2.0
Mode:2
```



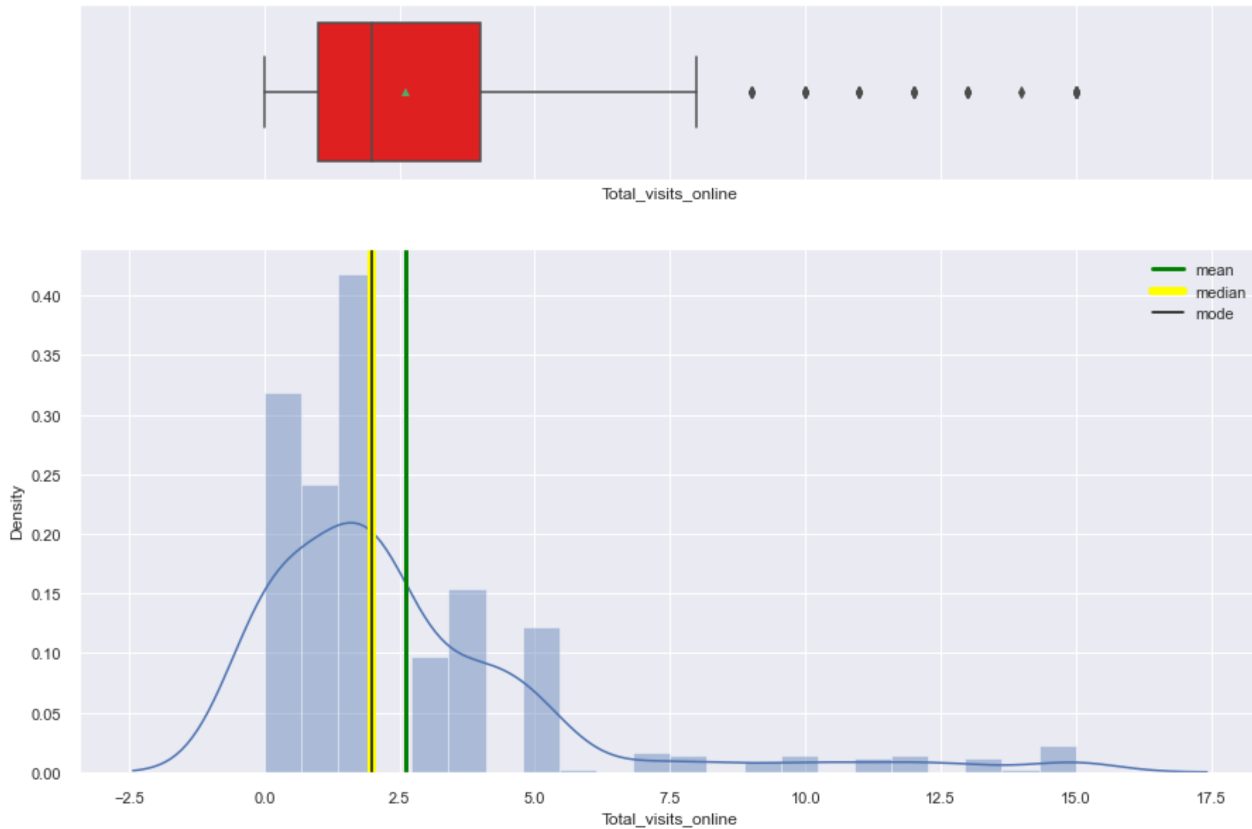

- Total_visits_bank seems fairly normal distributed.
- There are no outliers in the distribution.

## Observation on Total_visits_online

In [356…

```
histogram_boxplot(data.Total_visits_online)
```

```
Mean:2.6240369799691834
Median:2.0
Mode:2
```



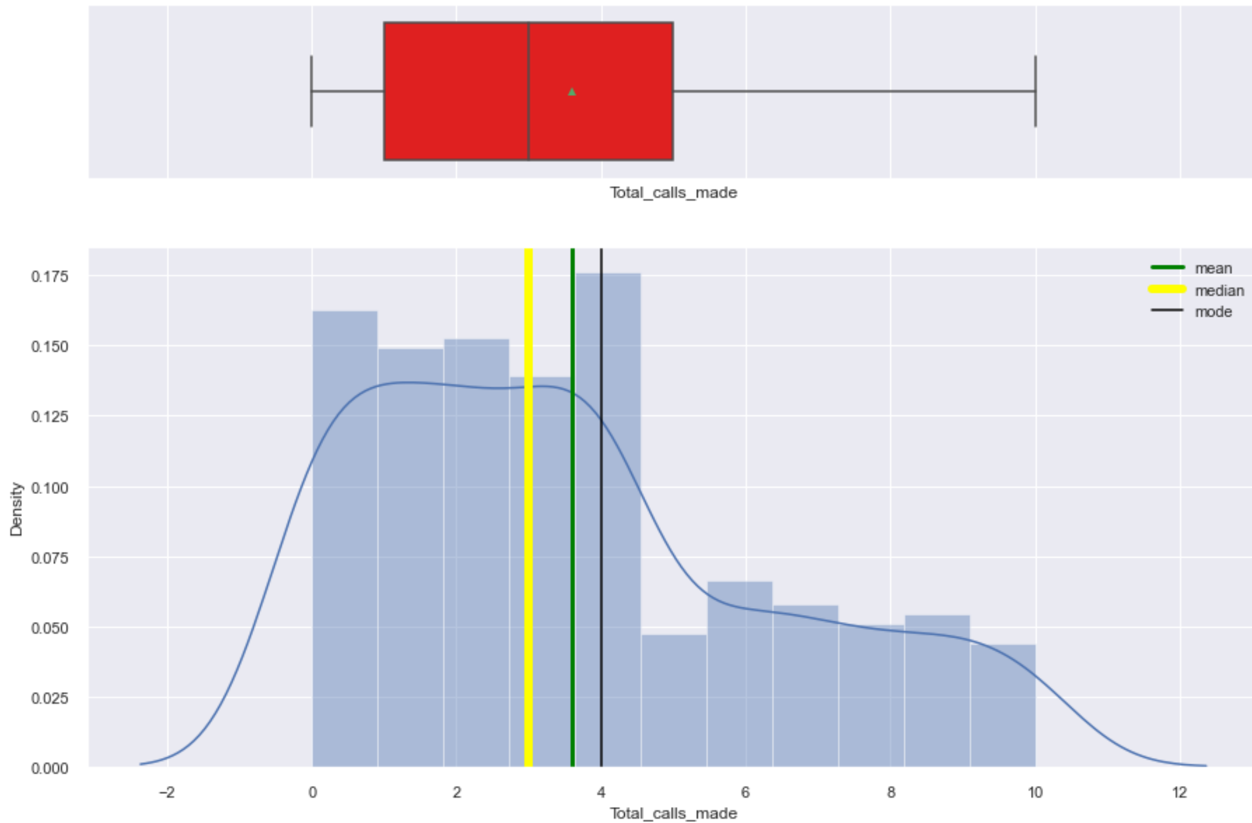

- Total_visits_online is right-skewed.
- There are many outliers to the right of the curve which may explain its skewness.

## Observation on Total_calls_made

In [357…

```
histogram_boxplot(data.Total_calls_made)
```

```
Mean:3.5901386748844377
Median:3.0
Mode:4
```



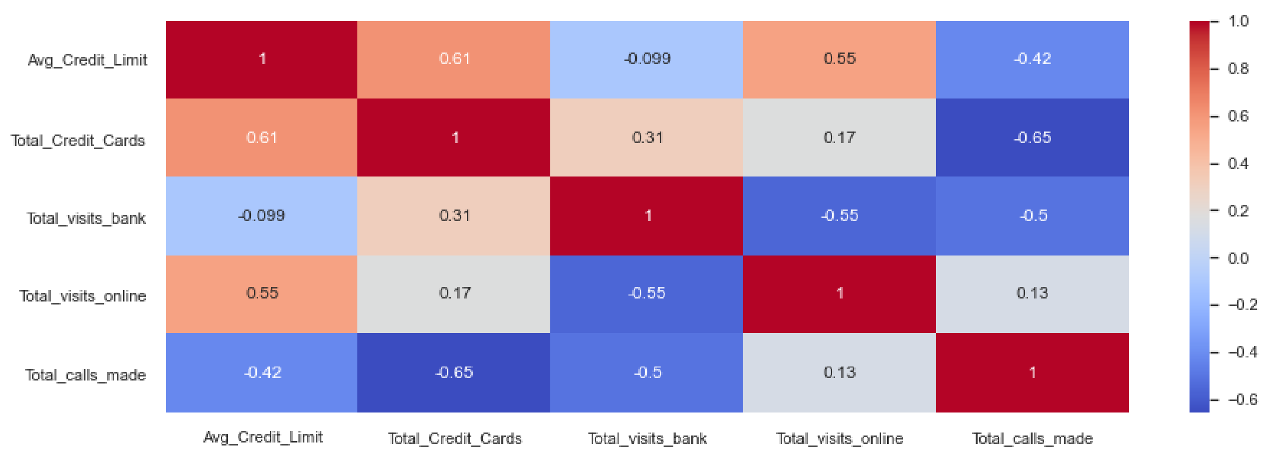

- Total_calls_made is right-skewed.
- There are no outliers.

In [ ]:

# Bivariate Analysis

In [358…

```
plt.figure(figsize=(15,5))
sns.heatmap(data.corr(),annot=True, cmap="coolwarm")
plt.show()
```




- The pairs that have a positive/high (>0.5) correlation are:
  - Total_Credit_Cards/Avg_Credit_Limit
  - Total_visits_online/Avg_Credit_Limit

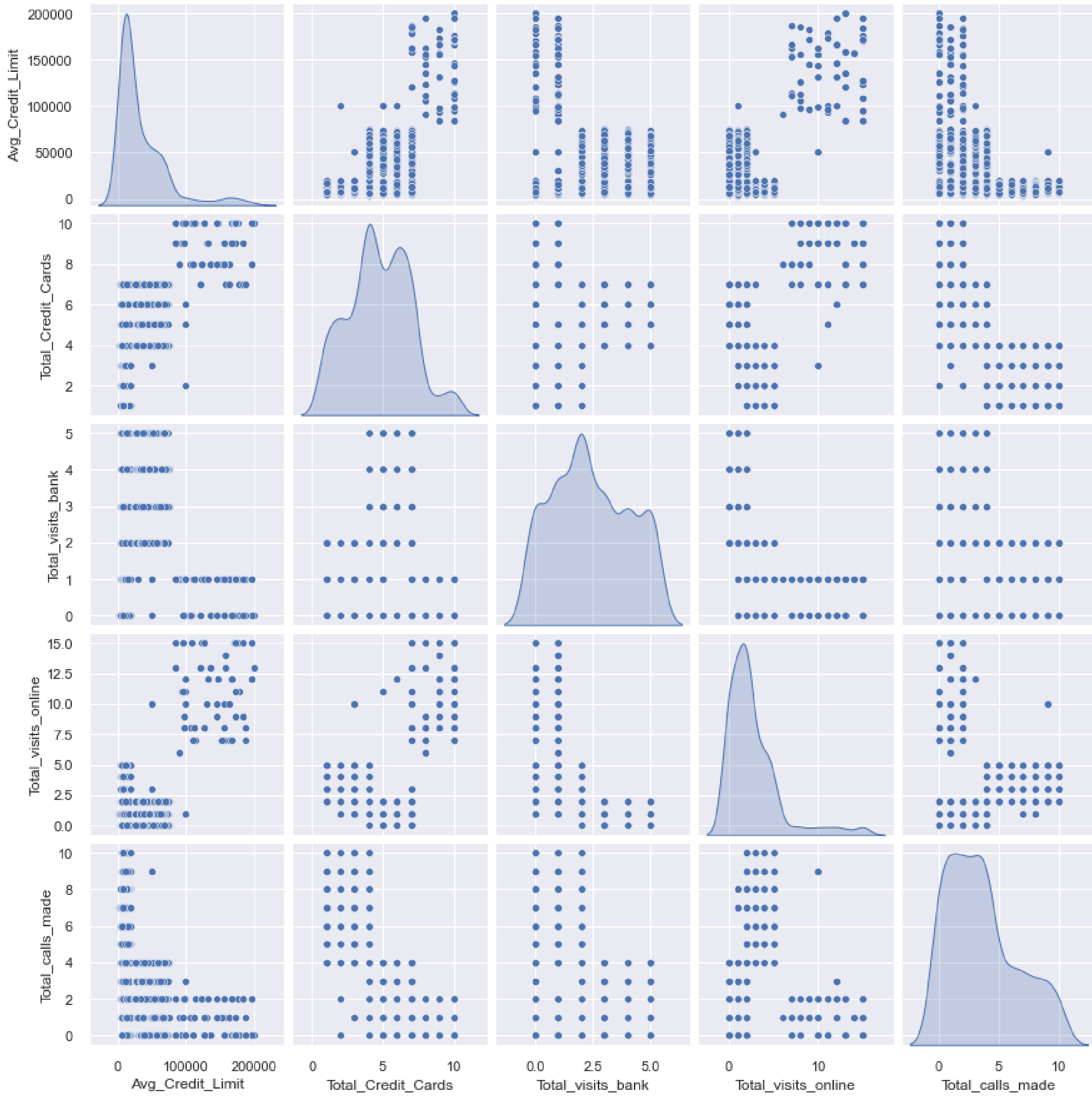
- The pairs that have a negative/high (<-0.5) correlation are:
    - Total_Credit_Cards/Total_calls_made
    - Total_visits_bank/Total_visits_online
    - Total_visits_bank/Total_calls_made

In [359…

```
sns.pairplot(data[all_col],diag_kind="kde")
```

Out[359…

```
<seaborn.axisgrid.PairGrid at 0x2a08e1db160>
```




**Observations**

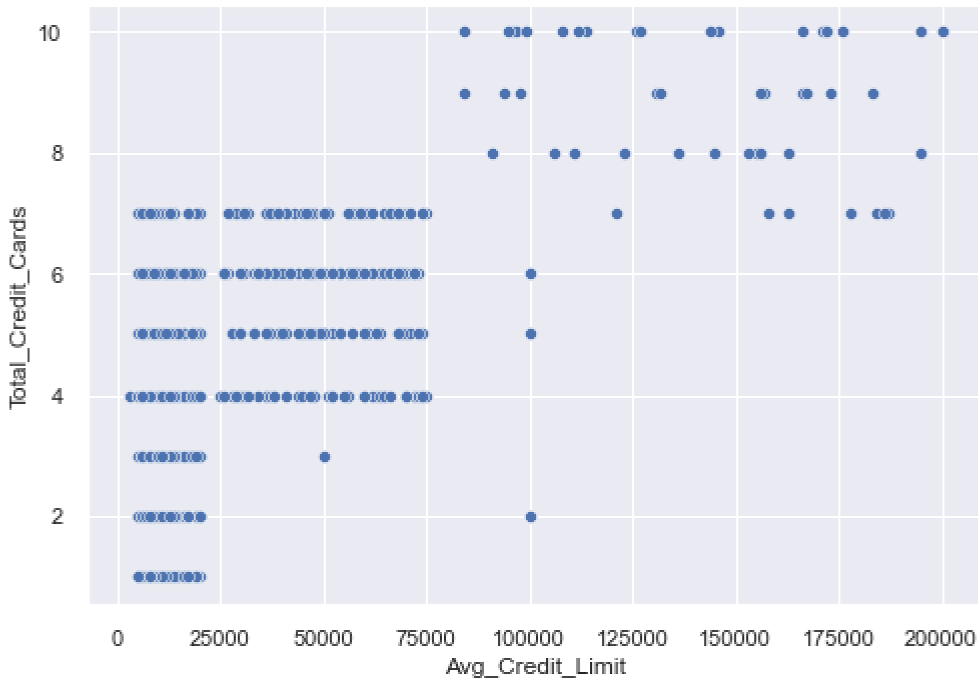
- There is no clear normally distributed feature
- Avg_Credit_limit and Total_visits_online are right-skewed
- Total_Credit_Cards and Total_calls_made seem multi-modal

In [ ]:

## Avg_Credit_Limit vs Total_Credit_Cards

In [360…
```
sns.scatterplot(x=data.Avg_Credit_Limit , y=data.Total_Credit_Cards)
```

Out[360…
```
<AxesSubplot:xlabel='Avg_Credit_Limit', ylabel='Total_Credit_Cards'>
```

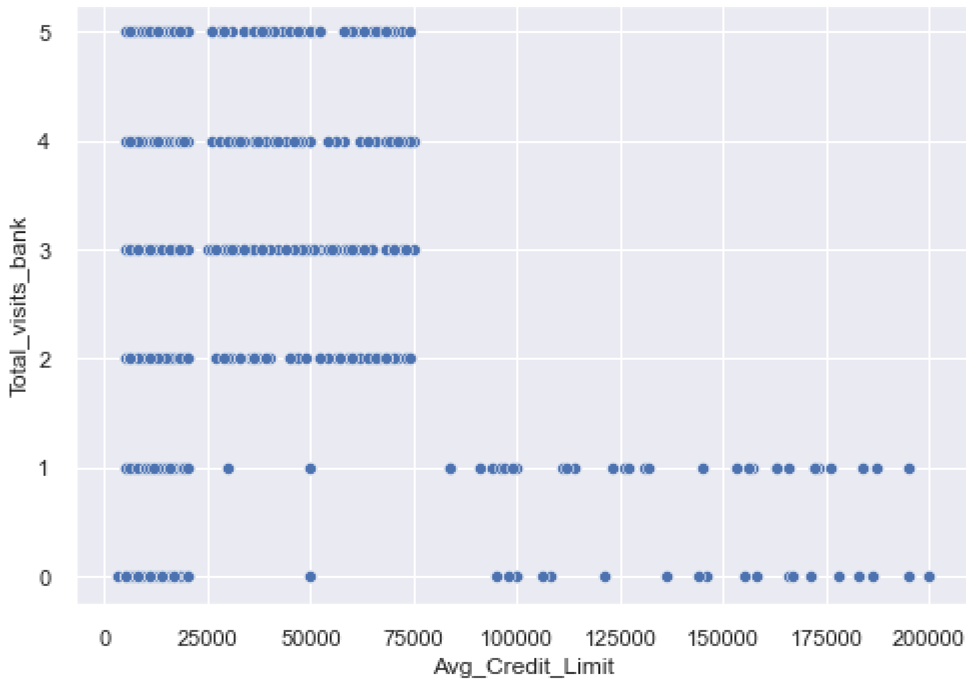



- It looks like those customers that have 7 or less Total_credit_cards have an Avg_Credit_limit of 75000 or less.
- As opposed to those that have a Total_Credit_Cards of 8 and above have an Avg_Credit_Limit above 75000.

## Avg_Credit_Limit vs Total_visits_bank

In [361…
```
sns.scatterplot(x=data.Avg_Credit_Limit , y=data.Total_visits_bank)
```

Out[361…
```
<AxesSubplot:xlabel='Avg_Credit_Limit', ylabel='Total_visits_bank'>
```




- It looks like those customers that have 1 or 0 Total_visits_bank have an Avg_Credit_limit either 25000 or below OR 85000 and above.
- Those customers that have Total_visits_bank 2 and above have an Avg_Credit_Limit of 75000 or less.

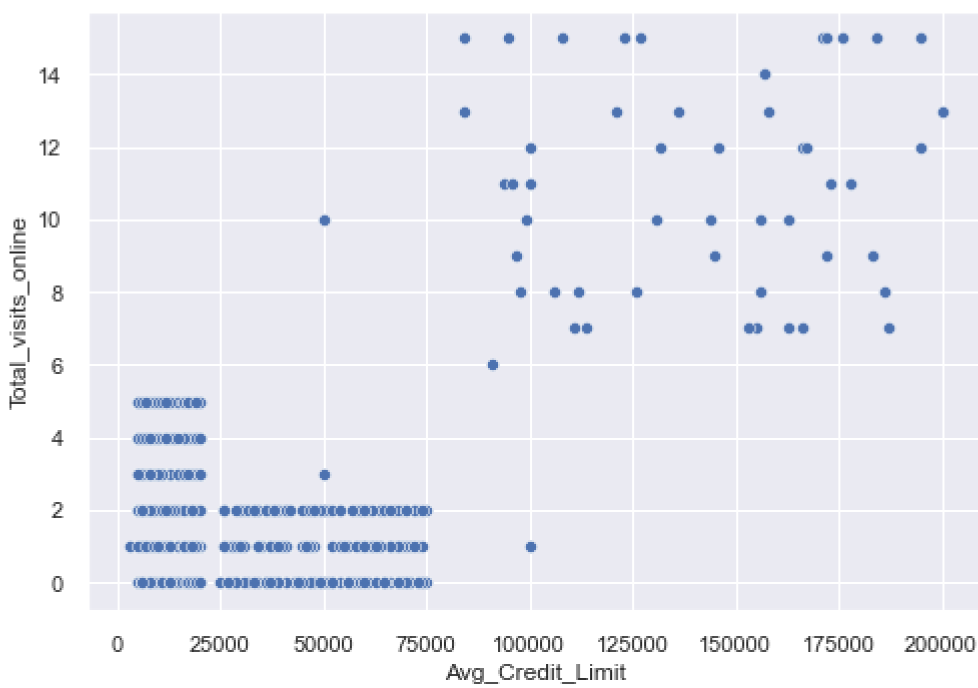
## Avg_Credit_Limit vs Total_visits_online

In [362…

```
sns.scatterplot(x=data.Avg_Credit_Limit , y=data.Total_visits_online)
```

Out[362…

```
<AxesSubplot:xlabel='Avg_Credit_Limit', ylabel='Total_visits_online'>
```



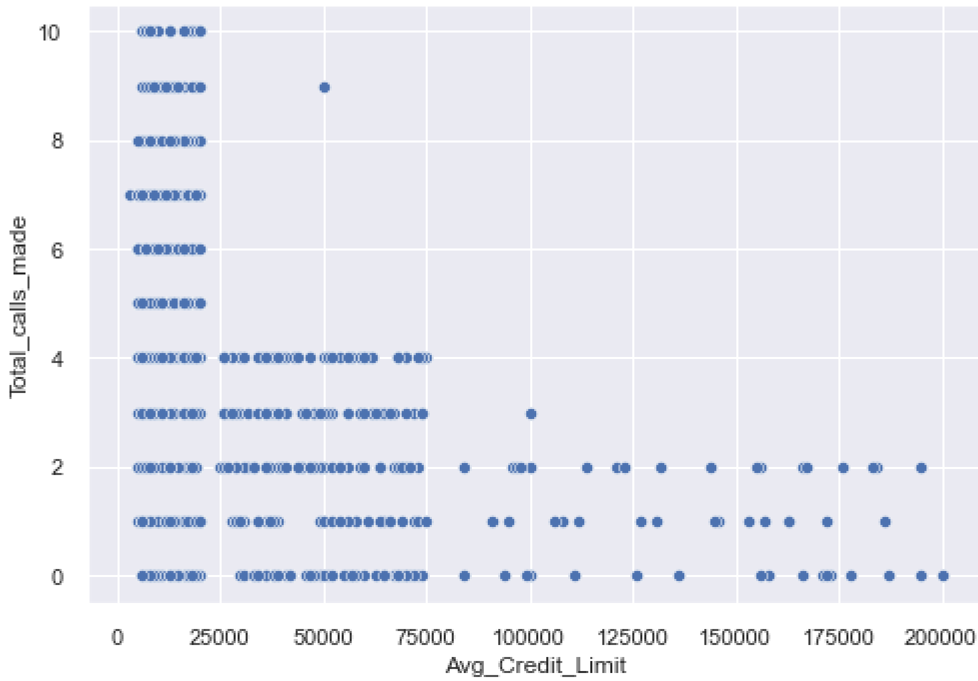

- It looks like those customers that have 6 or less Total_visits_online have an Avg_Credit_limit below 75000.

- While those customers that have Total_visits_online above 6, have an Avg_Credit_Limit above 75000.

## Avg_Credit_Limit vs Total_calls_made

In [363…
```
sns.scatterplot(x=data.Avg_Credit_Limit , y=data.Total_calls_made)
```

Out[363…
```
<AxesSubplot:xlabel='Avg_Credit_Limit', ylabel='Total_calls_made'>
```

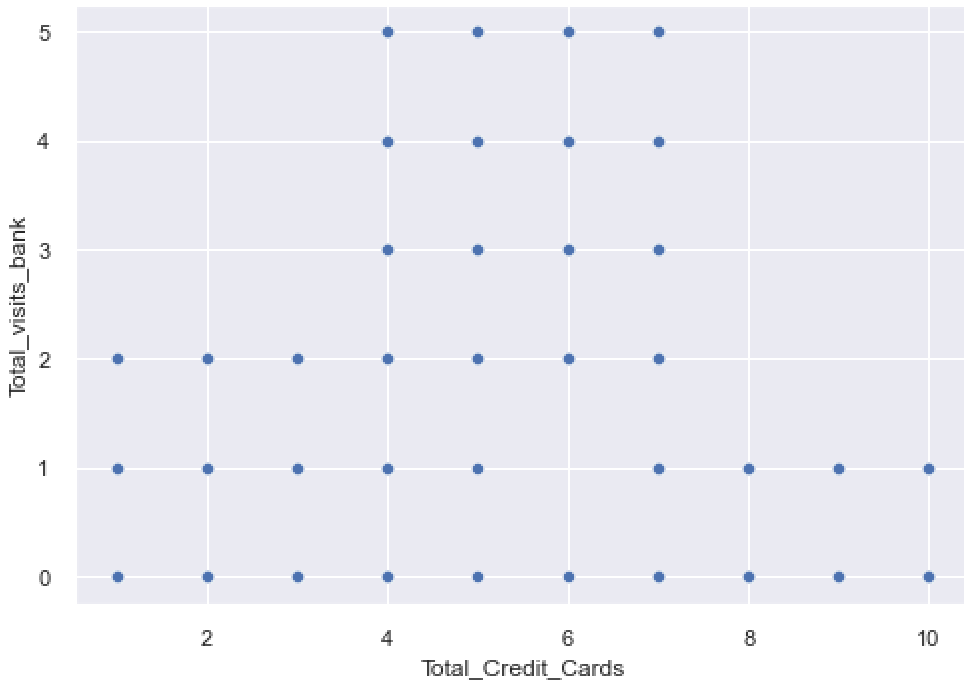



- It looks like those customers that have 5 or more Total_calls_made have an Avg_Credit_limit below 20000.
- Those customers that have 3 or 4 more Total_calls_made have an Avg_Credit_limit below 75000.
- Customers that have Total_calls_made 2 and below, have an Avg_Credit_Limit 200000 and below.

## Total_Credit_Cards vs Total_visits_bank

In [364…
```
sns.scatterplot(x=data.Total_Credit_Cards , y=data.Total_visits_bank)
```

Out[364…
```
<AxesSubplot:xlabel='Total_Credit_Cards', ylabel='Total_visits_bank'>
```

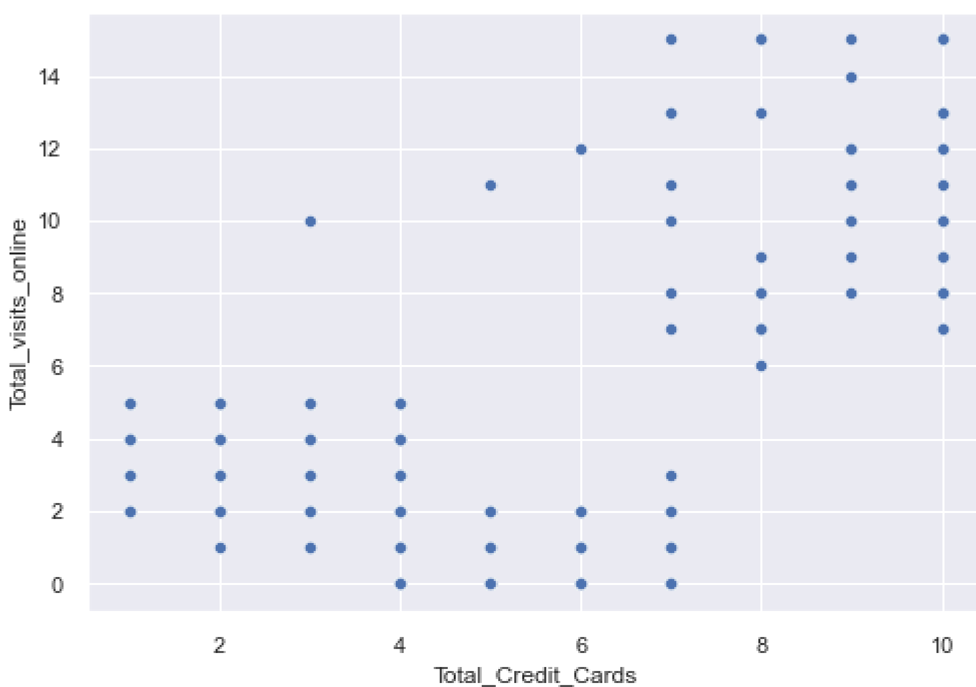



- Customers that have 3,4,5 Total_visits_bank have an Total_Credit_Cards between 4 and 7.
- Customers that have 2 Total_visits_bank have an Total_Credit_Cards between 1 and 7.
- Customers that have 0,1 Total_visits_bank have an Total_Credit_Cards between 1 and 10.

## Total_Credit_Cards vs Total_visits_online

In [365…
```
sns.scatterplot(x=data.Total_Credit_Cards , y=data.Total_visits_online)
```

Out[365…
```
<AxesSubplot:xlabel='Total_Credit_Cards', ylabel='Total_visits_online'>
```



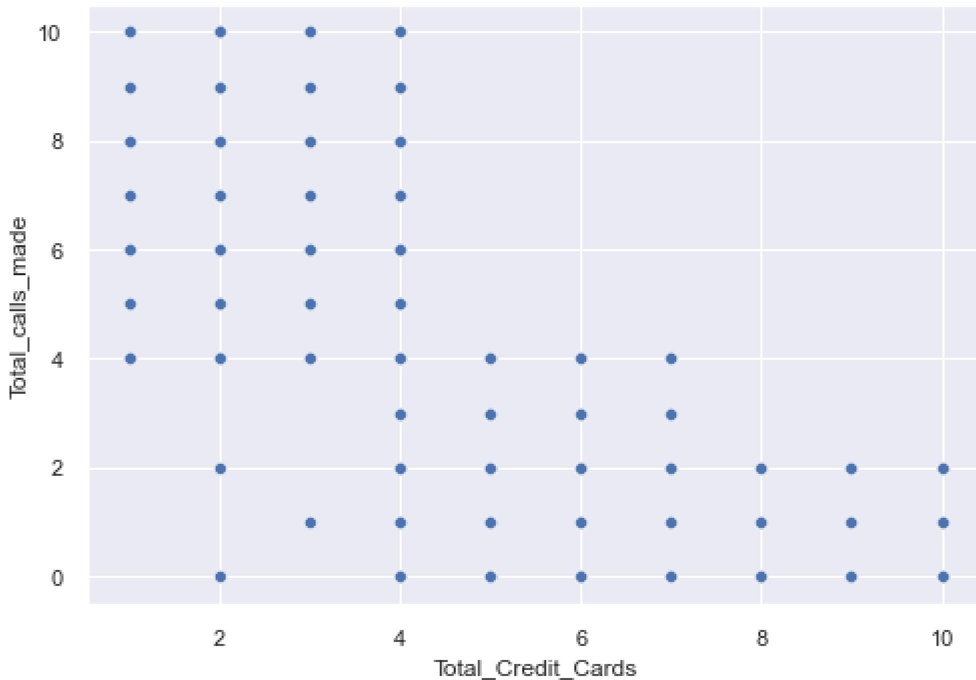

- Customers that have above 6 Total_visits_online have an Total_Credit_Cards between 7 and 10.
- Customers that have 6 or below Total_visits_online have an Total_Credit_Cards 7 and below.

## Total_Credit_Cards vs Total_calls_made

In [366...

```
sns.scatterplot(x=data.Total_Credit_Cards , y=data.Total_calls_made)
```

Out[366... <AxesSubplot:xlabel='Total_Credit_Cards', ylabel='Total_calls_made'>



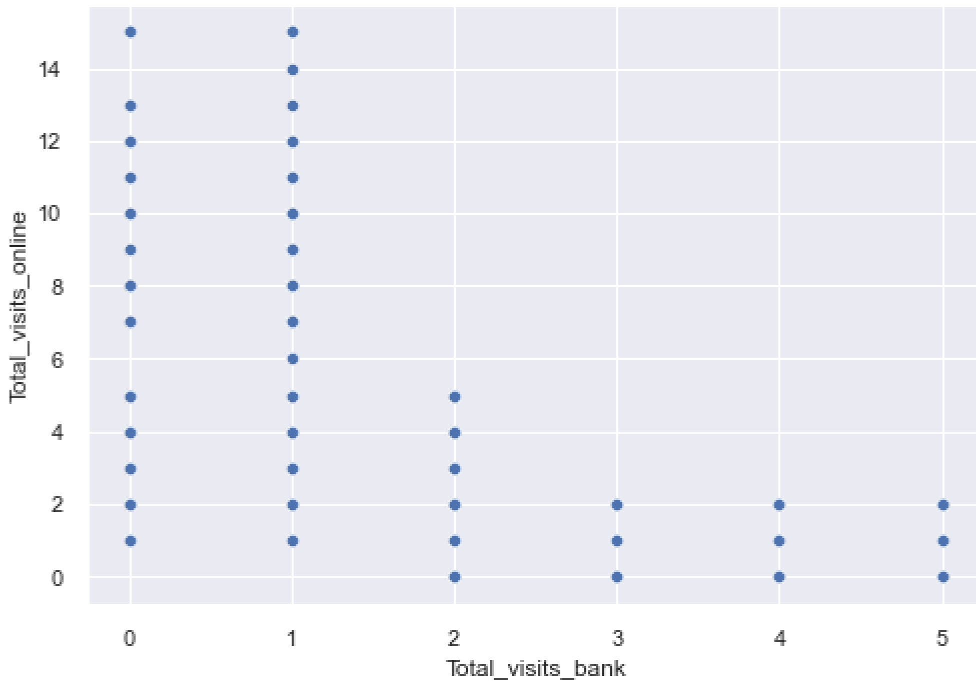

- Customers that have 4 and above Total_calls_made have an Total_Credit_Cards between 1 and 4.
- Customers that have 4 and below Total_calls_made have an Total_Credit_Cards 4 and 10.

## Total_visits_bank vs Total_visits_online

In [367...

```
sns.scatterplot(x=data.Total_visits_bank , y=data.Total_visits_online)
```

Out[367... <AxesSubplot:xlabel='Total_visits_bank', ylabel='Total_visits_online'>




- Customers that have 7 and above Total_visits_online have an Total_visits_bank between 0 and 1.
- Customers that have 6 and below Total_visits_online have an Total_visits_bank between 0 and 5.

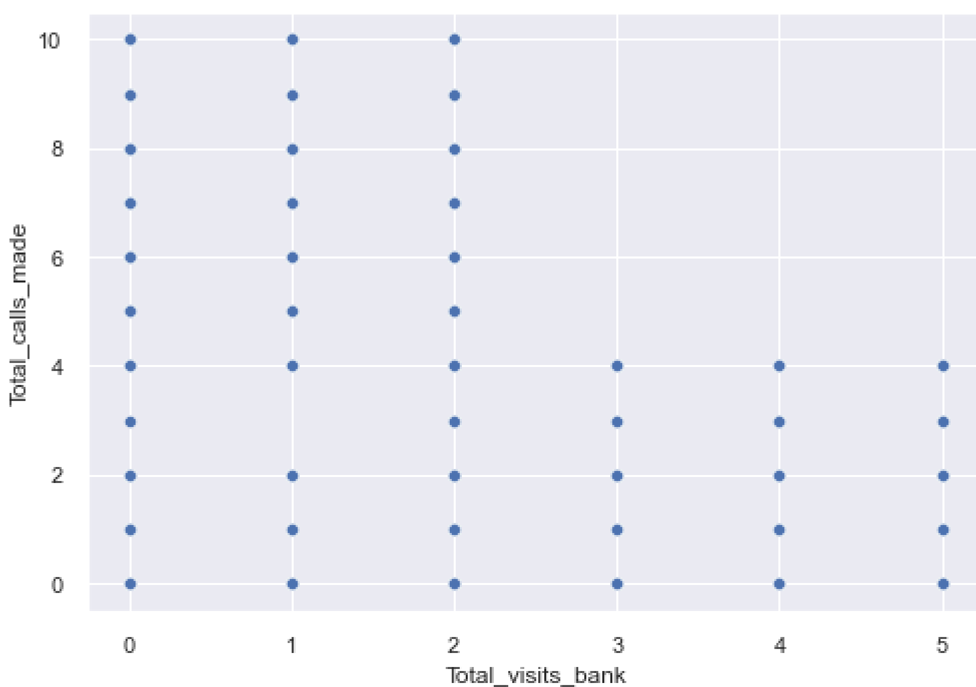
## Total_visits_bank vs Total_calls_made

In [368…

```
sns.scatterplot(x=data.Total_visits_bank , y=data.Total_calls_made)
```

Out[368…

```
<AxesSubplot:xlabel='Total_visits_bank', ylabel='Total_calls_made'>
```




- Customers that have 5 and above Total_calls_made have an Total_visits_bank between 0 and 2.
- Customers that have 4 and below Total_calls_made have an Total_visits_bank between 0 and 5.

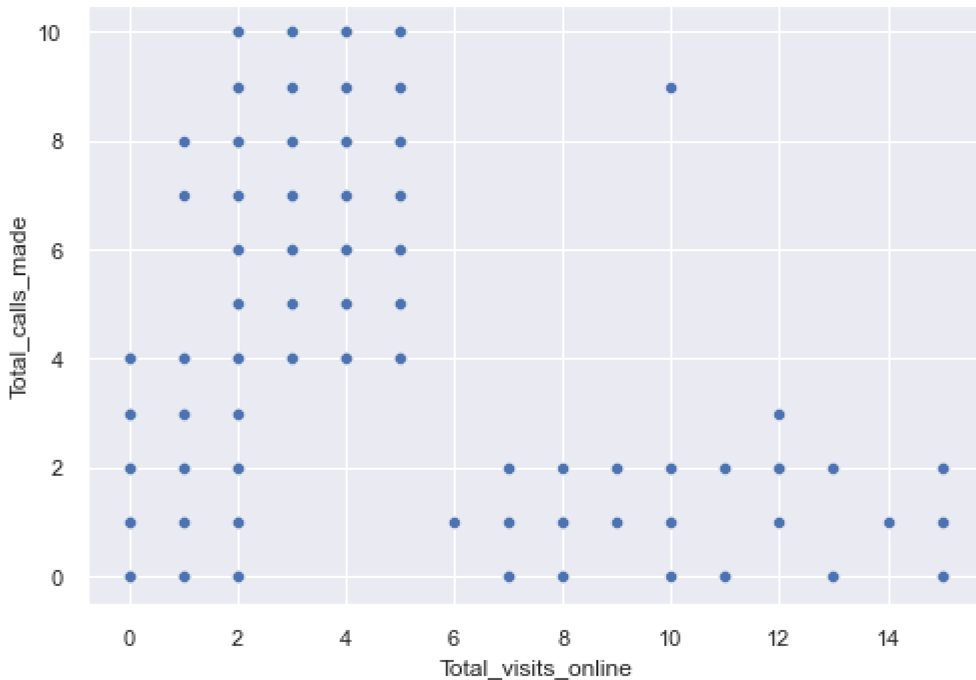
### Total_visits_online vs Total_calls_made

In [369…

```
sns.scatterplot(x=data.Total_visits_online , y=data.Total_calls_made)
```

Out[369…

```
<AxesSubplot:xlabel='Total_visits_online', ylabel='Total_calls_made'>
```




- Customers that have 3 and above Total_calls_made have an Total_visits_online between 4 and 10.
- Customers that have 3 and below Total_calls_made have an Total_visits_online between 0 and 15.

### Key Meaningful Observations

- It looks like there is exclusive relationship between `Total_visits_bank` vs `Total_visits_online` vs `Total_calls_made` . Where customers that are predominant in one variable are weak in the others.
- `Avg_Credit_limit` seems to be divided in groups of those that have 75000 and below and those that have 75000 and above. This variable has the most outliers.

In [ ]:

# Data Pre-processing

### Outlier treatment

- We will not treat outliers in this case.
- `Avg_Credit_limit` as the most outliers, but in this case it seems that rows with 75000 and above could be included in its own cluster. We will not treat outliers.

## Missing-Value Treatment

- All the features are numeric, and we have already treated duplicates and there are no null values.There is no need to treat missing values. as this is not applicable in this case.

## Scaling the Dataset

In [370…

```
all_col = data.select_dtypes(include=np.number).columns.tolist()

scaler=StandardScaler()
subset=data[all_col].copy()
subset_scaled=scaler.fit_transform(subset)
subset_scaled_df=pd.DataFrame(subset_scaled,columns=subset.columns)
```

# Applying K-means clustering algorithms

## Elbow curve

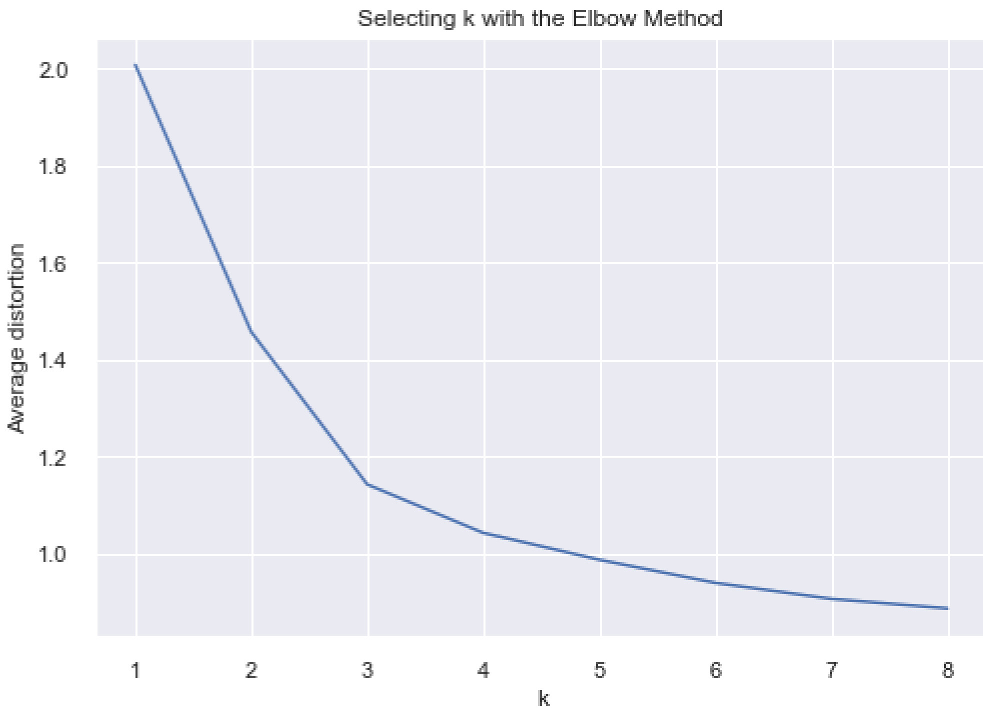
In [371…

```
clusters=range(1,9)
meanDistortions=[]

for k in clusters:
    model=KMeans(n_clusters=k)
    model.fit(subset_scaled_df)
    prediction=model.predict(subset_scaled_df)
    distortion=sum(np.min(cdist(subset_scaled_df, model.cluster_centers_, 'euclidean'),

    meanDistortions.append(distortion)

    print('Number of Clusters:', k, '\tAverage Distortion:', distortion)

plt.plot(clusters, meanDistortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
```

```
Number of Clusters: 1   Average Distortion: 2.007896349270688
Number of Clusters: 2   Average Distortion: 1.4576197022077821
Number of Clusters: 3   Average Distortion: 1.1434401208195095
Number of Clusters: 4   Average Distortion: 1.0435538595477063
Number of Clusters: 5   Average Distortion: 0.9880591433704322
Number of Clusters: 6   Average Distortion: 0.9404952836425913
Number of Clusters: 7   Average Distortion: 0.9075861543551181
Number of Clusters: 8   Average Distortion: 0.887984835729803
```

Out[371…

```
Text(0.5, 1.0, 'Selecting k with the Elbow Method')
```



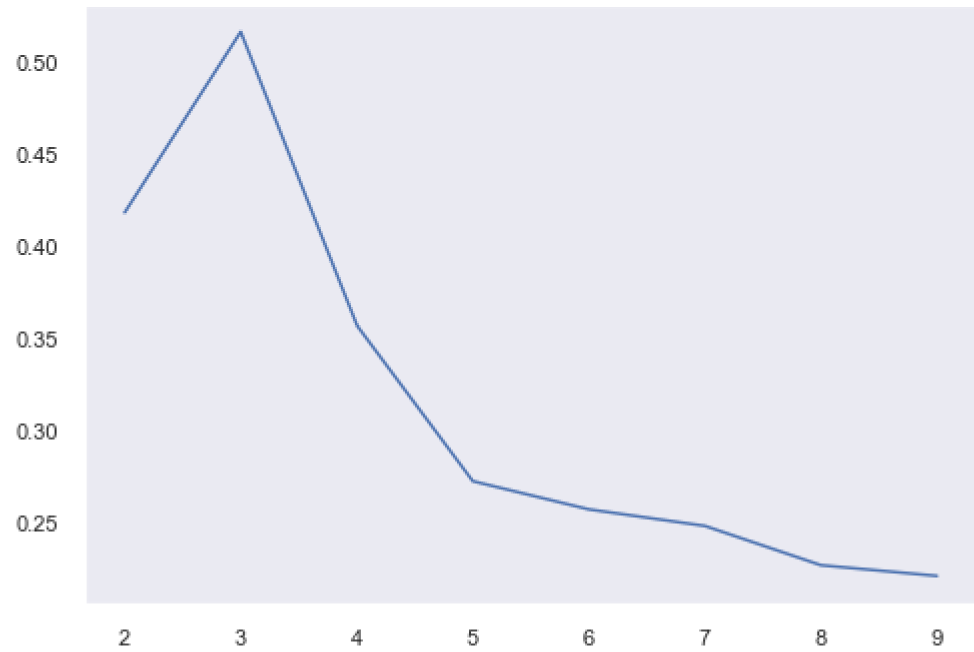

- Appropriate k seems to be 3 or 4.

## Silhouette Score

In [372...

```
sil_score = []
cluster_list = list(range(2,10))
for n_clusters in cluster_list:
    clusterer = KMeans(n_clusters=n_clusters)
    preds = clusterer.fit_predict((subset_scaled_df))
    #centers = clusterer.cluster_centers_
    score = silhouette_score(subset_scaled_df, preds)
    sil_score.append(score)
    print("For n_clusters = {}, silhouette score is {})".format(n_clusters, score))

plt.plot(cluster_list,sil_score)
plt.grid()
```

```
For n_clusters = 2, silhouette score is 0.41800025566689647)
For n_clusters = 3, silhouette score is 0.516281010855363)
For n_clusters = 4, silhouette score is 0.3570238219413198)
For n_clusters = 5, silhouette score is 0.2722848313346344)
For n_clusters = 6, silhouette score is 0.25696498143767876)
For n_clusters = 7, silhouette score is 0.24796181778236623)
For n_clusters = 8, silhouette score is 0.22660108820428626)
For n_clusters = 9, silhouette score is 0.22077645663369874)
```



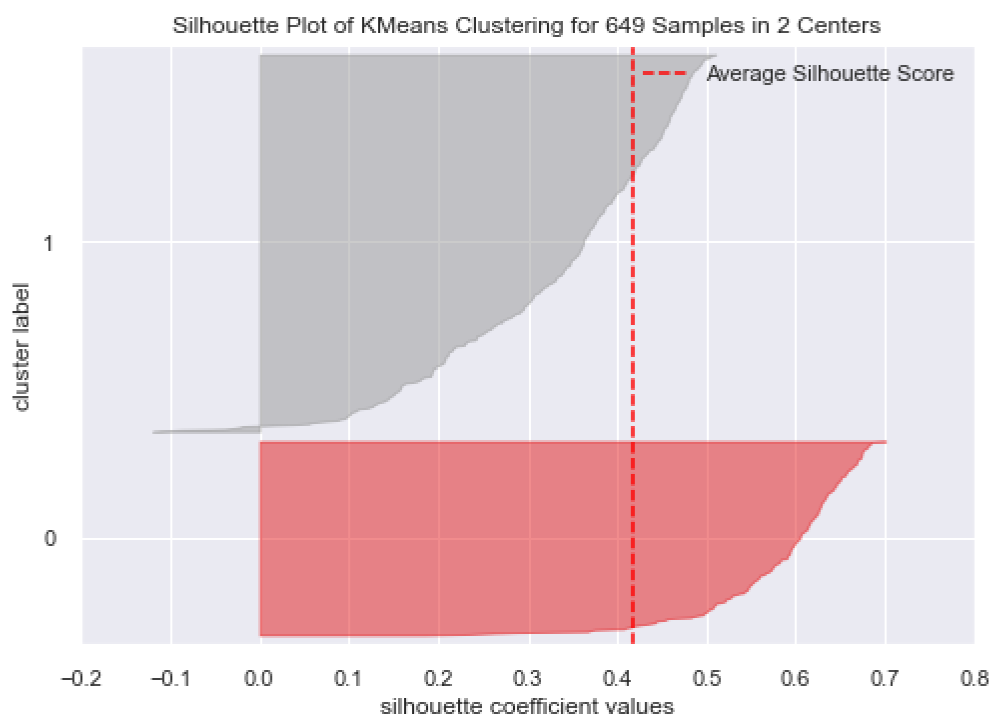

- Silhouette score for 3 is high (0.5162), so we will choose 3 as value of k.

## Appropriate number of cluster with silhouette coefficients

### Visualizing with 2 Clusters

In [373...

```
visualizer = SilhouetteVisualizer(KMeans(2, random_state = 1))
visualizer.fit(subset_scaled_df)
visualizer.show();
```

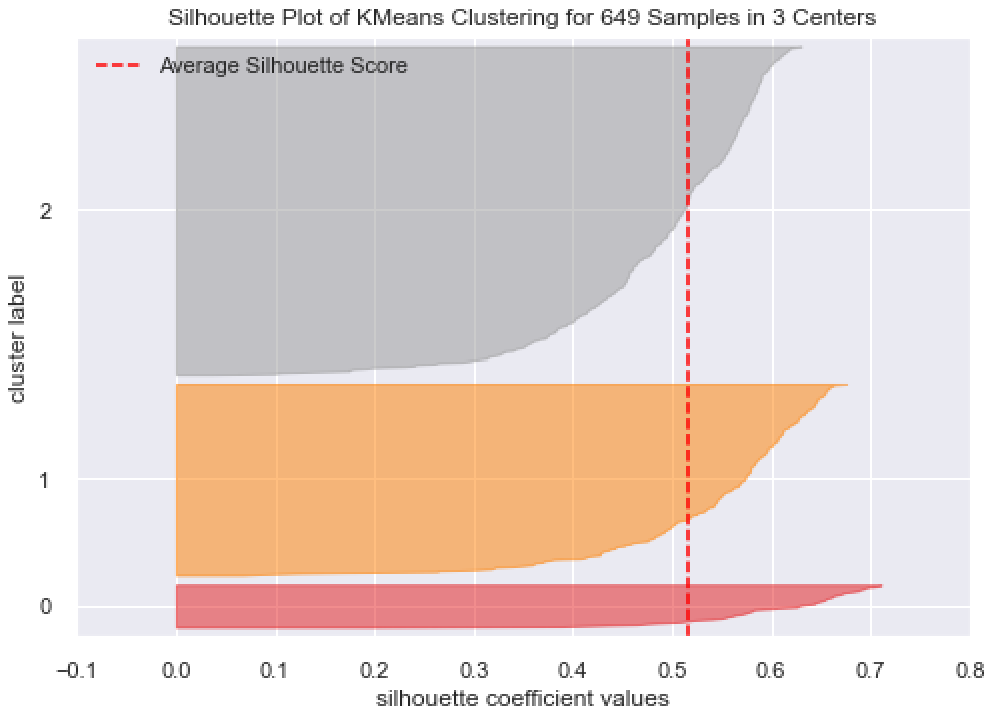



- Cluster-1 looks like it is overfitting on some data points.
- Cluster-0 seems far from the mean value.

### Visualizing with 3 Clusters

In [374...

```
visualizer = SilhouetteVisualizer(KMeans(3, random_state = 1))
visualizer.fit(subset_scaled_df)
visualizer.show();
```

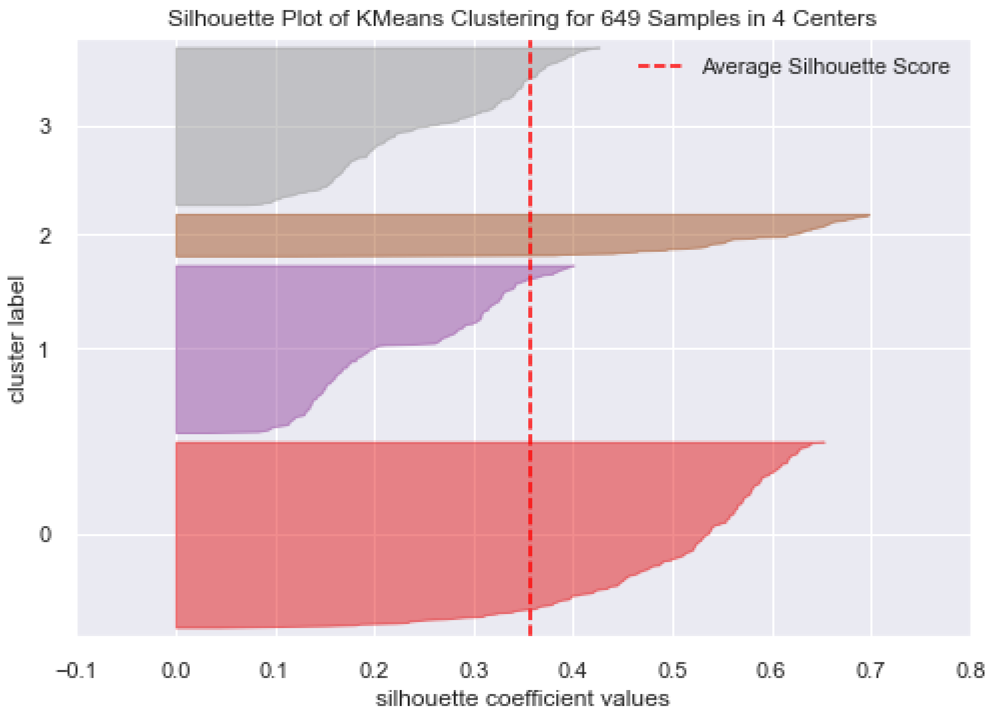



- None of the clusters are overfitting.
- All clusters seem to be at the same distince from the mean value.

### Visualizing with 4 Clusters

In [375...

```
visualizer = SilhouetteVisualizer(KMeans(4, random_state = 1))
visualizer.fit(subset_scaled_df)
visualizer.show();
```




- None of the clusters are overfitting.
- Clusters 3 and 1 are close to the mean value, while clusters 2 and 0 are far from the mean.

### 3 is the appropriate amount of clusters as silhoutte score is high enough and there is a change in direction in elbow curve for this number.

In [376…

```
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(subset_scaled_df)
```

Out[376… KMeans(n_clusters=3, random_state=0)

In [377…

```
data['K_means_segments'] = kmeans.labels_
#subset_scaled_df['K_means_segments'] = kmeans.labels_

cluster_profile = data.groupby('K_means_segments').mean()
cluster_profile['count_in_each_segment'] = data.groupby('K_means_segments')['Avg_Credit

cluster_profile
```

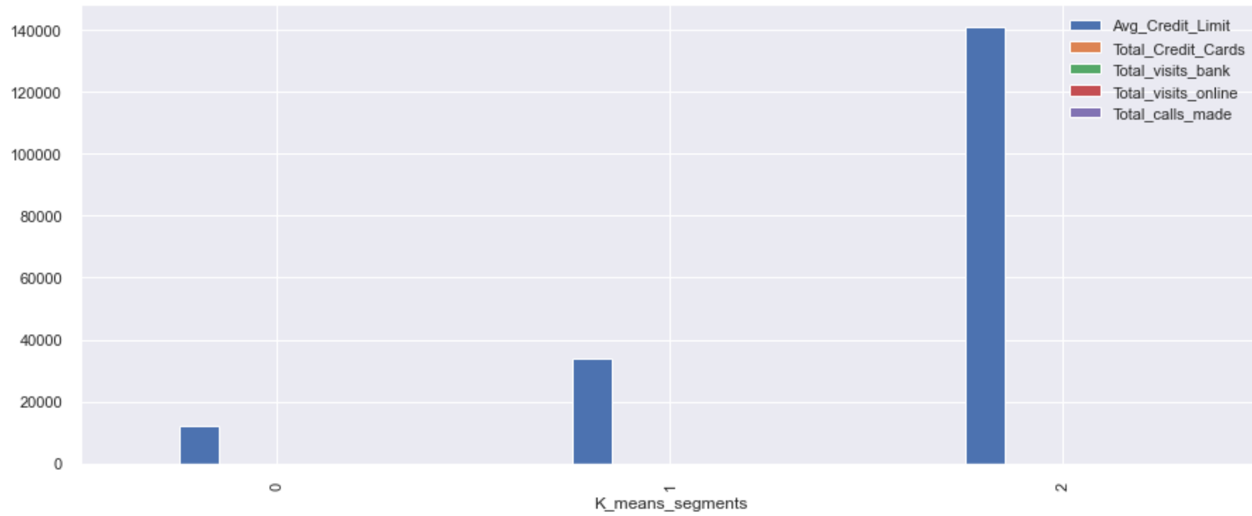
Out[377…

| K_means_segments | Avg_Credit_Limit | Total_Credit_Cards | Total_visits_bank | Total_visits_online | Total_calls_ |
|---|---|---|---|---|---|
| **0** | 12239.819005 | 2.411765 | 0.945701 | 3.561086 | 6.8[illegible] |
| **1** | 34071.428571 | 5.518519 | 3.484127 | 0.981481 | 1.9[illegible] |
| **2** | 141040.000000 | 8.740000 | 0.600000 | 10.900000 | 1.0[illegible] |

In [378…

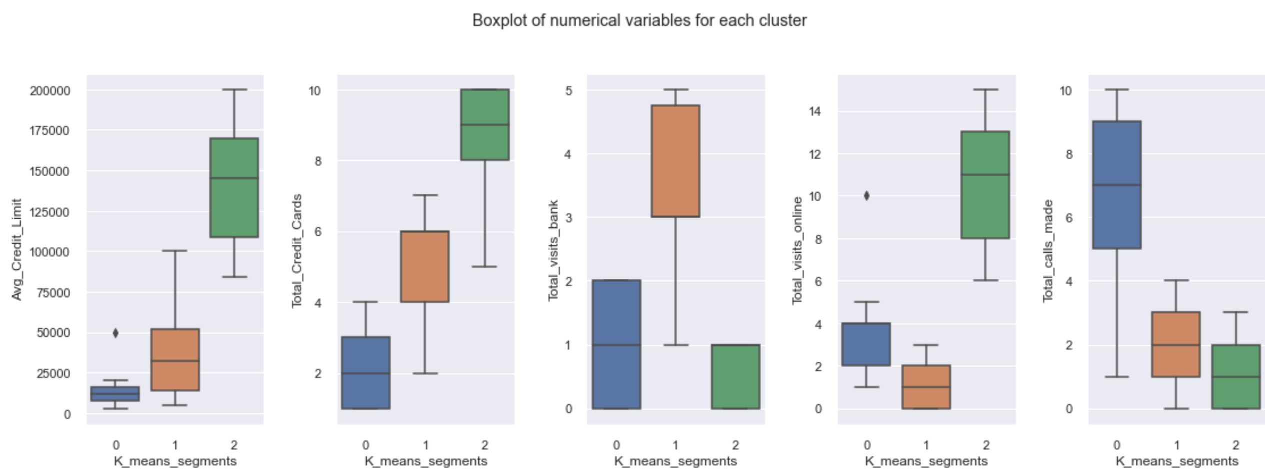
```
data.groupby('K_means_segments').mean().plot.bar(figsize=(15,6))
```

Out[378… <AxesSubplot:xlabel='K_means_segments'>




In [379...

```
fig, axes = plt.subplots(1, 5,  figsize=(16, 6))
fig.suptitle('Boxplot of numerical variables for each cluster')
counter = 0
for ii in range(5):
    sns.boxplot(ax=axes[ii],y=data[all_col[counter]],x=data['K_means_segments'])
    counter = counter+1

fig.tight_layout(pad=2.0)
```




## Insights - K-Means clustering

- **Cluster 0** :
    - This cluster contains customers with low `Avg_Credit_Limit` (less than 25000)
    - Also, low `Total_Credit_Cards` (3 or less)
    - This cluster prefers to interact with the bank via phone, hence the `Total_calls_made` is high.
    - This cluster does not prefers to interact with the bank via bank visits, nor online.
- **Cluster 1**:
    - This cluster contains customers with average `Avg_Credit_Limit` (between 20000 and 60000)
    - Also, average `Total_Credit_Cards` (between 4 and 6)

- This cluster prefers to interact with the bank via bank visits, hence the `Total_visit_bank` is high.
- This cluster does not prefers to interact with the bank via phone, nor online.

- **Cluster 2**:

  - This cluster contains customers with high `Avg_Credit_Limit` (between 100000 and 175000)
  - Also, high `Total_Credit_Cards` (between 8 and 10)
  - This cluster prefers to interact with the bank via online, hence the `Total_calls_made` is high.
  - This cluster does not prefers to interact with the bank via bank visits, nor phone.

# Applying Hierarchical clustering

## Apply Hierarchical clustering with different linkage methods

In [380…

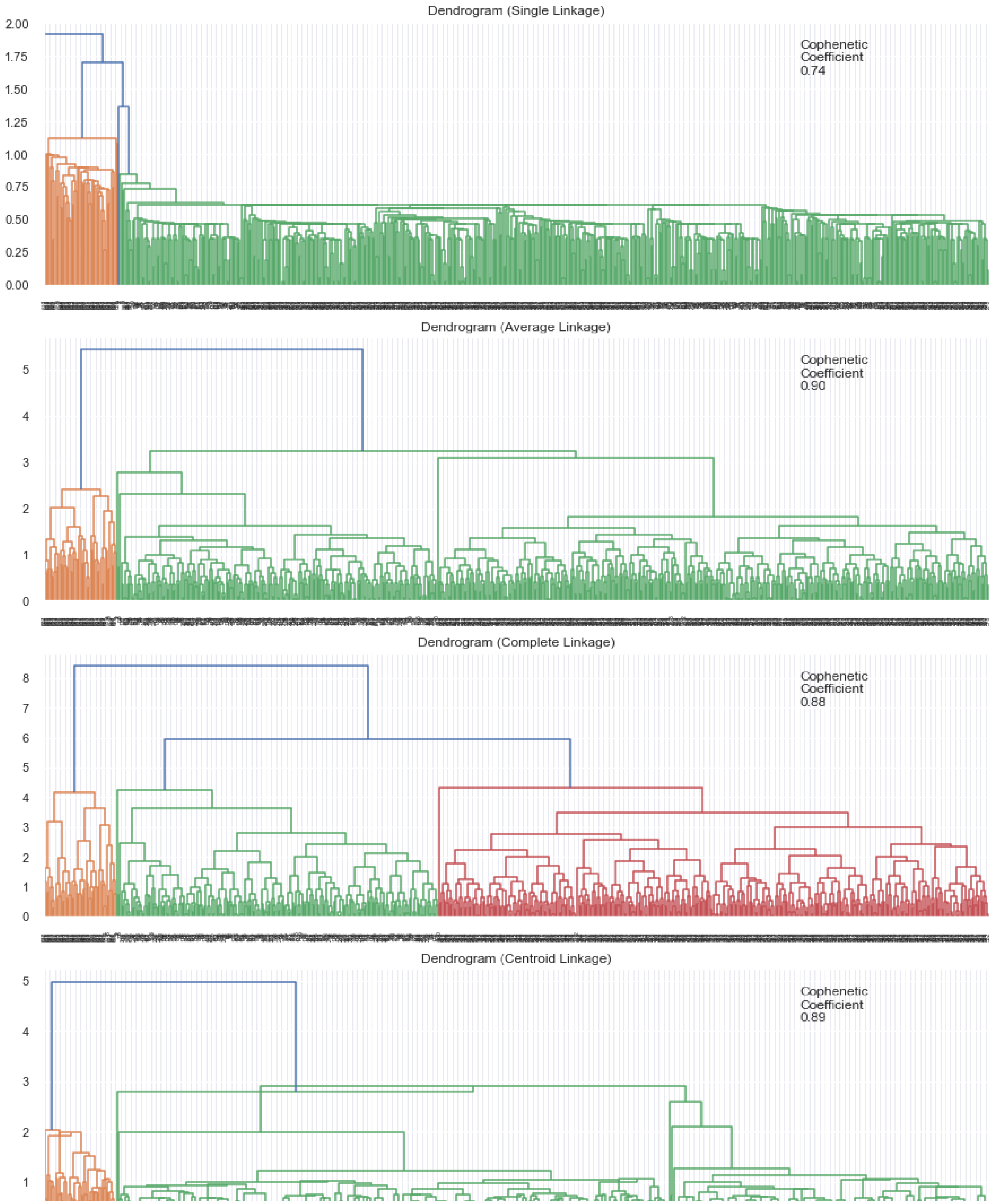
```
# cophenet index is a measure of the correlation between the distance of points in feat
# closer it is to 1, the better is the clustering

distance_metrics = [ 'euclidean']
linkage_methods = ['single', 'average', 'complete','centroid','ward','weighted']
high_cophenet_corr = 0
high_dm_lm = [0,0]
for dm in distance_metrics:
    for lm in linkage_methods:
        Z = linkage(subset_scaled_df, metric=dm, method=lm)
        c, coph_dists = cophenet(Z , pdist(subset_scaled_df))
        print('Cophenetic correlation for distance metrics {} and linkage method {} is
        if high_cophenet_corr < c:
            high_cophenet_corr = c
            high_dm_lm[0] = dm
            high_dm_lm[1] = lm
```

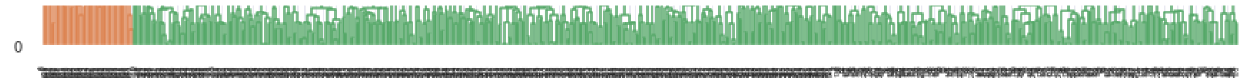
```
Cophenetic correlation for distance metrics euclidean and linkage method single is 0.739
5135051413775
Cophenetic correlation for distance metrics euclidean and linkage method average is 0.89
74425535306298
Cophenetic correlation for distance metrics euclidean and linkage method complete is 0.8
794736468795109
Cophenetic correlation for distance metrics euclidean and linkage method centroid is 0.8
94471288720818
Cophenetic correlation for distance metrics euclidean and linkage method ward is 0.74258
13590948763
Cophenetic correlation for distance metrics euclidean and linkage method weighted is 0.8
551098644586315
```

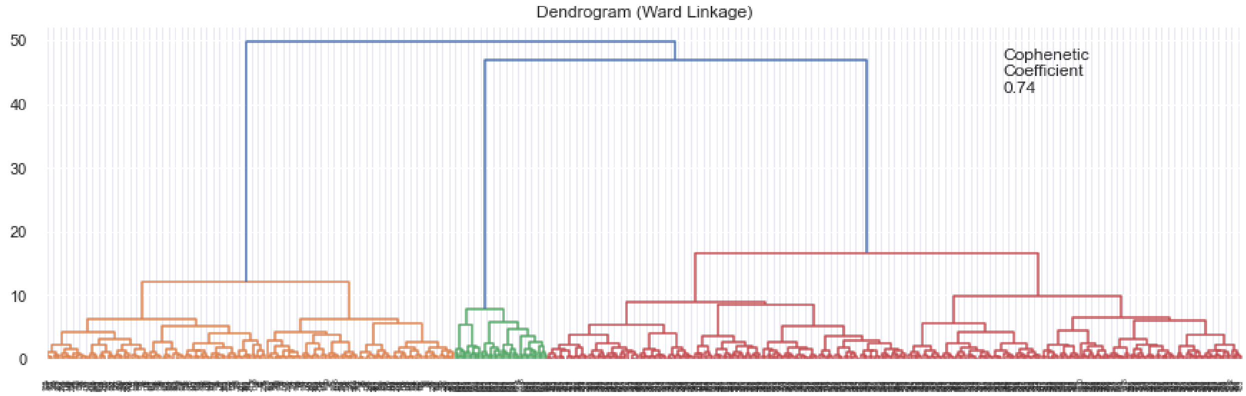
In [381…

```
# List of all linkage methods to check
methods = ['single',
           'average',
           'complete',
           'centroid',
           'ward',
          'weighted']

compare_cols = ['Linkage', 'Cophenetic Coefficient']

fig, axs = plt.subplots(len(methods), 1, figsize=(15, 30))
```

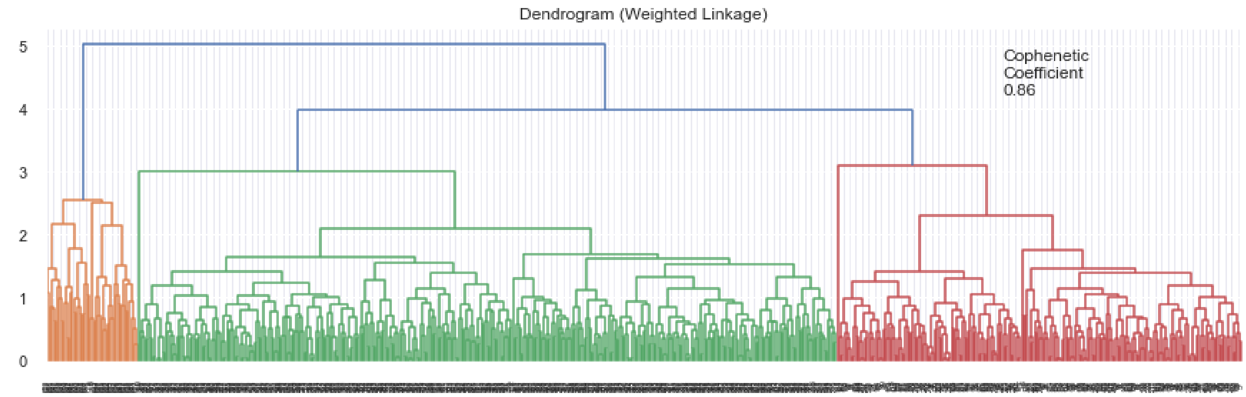
```python
# Enumerate through the list of all methods above
# Get linkage, plot dendrogram, calculate cophenetic coefficient
for i, method in enumerate(methods):

    Z = linkage(subset_scaled_df, metric='euclidean', method=method)

    dendrogram(Z, ax=axs[i]);
    axs[i].set_title(f'Dendrogram ({method.capitalize()} Linkage)')
    coph_corr, coph_dist = cophenet(Z, pdist(subset_scaled_df))
    axs[i].annotate(f'Cophenetic\nCoefficient\n{coph_corr:0.2f}',
                    (0.80, 0.80),
                    xycoords='axes fraction')
```













- Out of all the dendrogram we see, it is clear that dendrogram with ward linkage method gave us separate and distinct clusters(Cophenetic Coefficient 0.7576 ).
- Looks like 3 clusters would be appropriate number of cluster from dendrogram with Ward Linkage method

## Create 3 clusters

In [382...

```
#Trying with K value as 3
HCmodel = AgglomerativeClustering(n_clusters=3,affinity='euclidean', linkage='ward')
HCmodel.fit(subset_scaled_df)
subset_scaled_df['HC_Clusters'] = HCmodel.labels_

data['HC_Clusters'] = HCmodel.labels_

cluster_profile = data.groupby('HC_Clusters').mean()

cluster_profile['count_in_each_segments'] = data.groupby('HC_Clusters')['Avg_Credit_Lim
```

In [383...

```
# lets see names of Total_Credit_Cards in each cluster
for cl in data['HC_Clusters'].unique():
    print('In cluster ', cl ,' Total_Credit_Cards are: ')
    print(data[data['HC_Clusters']==cl]['Total_Credit_Cards'].unique())
```

```
In cluster  0  Total_Credit_Cards are:
[2 7 5 4 6]
In cluster  1  Total_Credit_Cards are:
[3 2 4 1 5]
In cluster  2  Total_Credit_Cards are:
[ 6  5  9  8 10  7]
```

In [384…

```
# lets see names of Total_visits_bank in each cluster
for cl in data['HC_Clusters'].unique():
    print('In cluster ', cl ,' Total_visits_bank are: ')
    print(data[data['HC_Clusters']==cl]['Total_visits_bank'].unique())
```

```
In cluster  0  Total_visits_bank are:
[1 2 5 3 4]
In cluster  1  Total_visits_bank are:
[0 2 1]
In cluster  2  Total_visits_bank are:
[0 1]
```

In [385…

```
# lets see names of Total_visits_online in each cluster
for cl in data['HC_Clusters'].unique():
    print('In cluster ', cl ,' Total_visits_online are: ')
    print(data[data['HC_Clusters']==cl]['Total_visits_online'].unique())
```

```
In cluster  0  Total_visits_online are:
[1 3 0 2]
In cluster  1  Total_visits_online are:
[10  1  2  5  4  3]
In cluster  2  Total_visits_online are:
[12 11 14  7 10 13 15  6  8  9]
```

In [386…

```
# lets see names of Total_calls_made in each cluster
for cl in data['HC_Clusters'].unique():
    print('In cluster ', cl ,' Total_calls_made are: ')
    print(data[data['HC_Clusters']==cl]['Total_calls_made'].unique())
```

```
In cluster  0  Total_calls_made are:
[0 4 2 3 1]
In cluster  1  Total_calls_made are:
[ 9  8  1  2  7  5  6  4 10]
In cluster  2  Total_calls_made are:
[3 2 1 0]
```

## Display Cluster Profile

In [387…

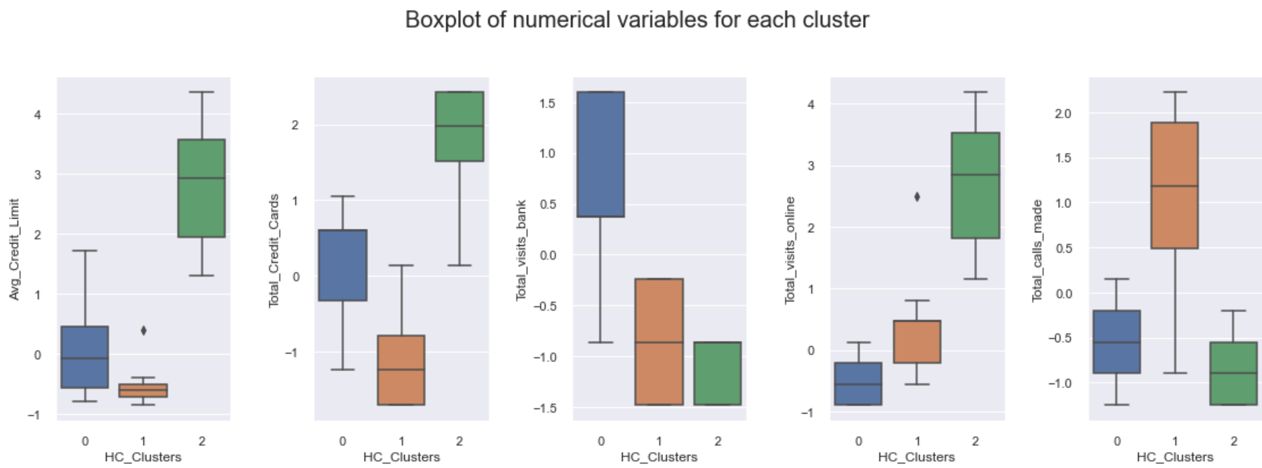
```
cluster_profile.style.highlight_max(color = 'lightgreen', axis = 0)
```

Out[387…

| HC_Clusters | Avg_Credit_Limit | Total_Credit_Cards | Total_visits_bank | Total_visits_online | Total_calls_made |
|---|---|---|---|---|---|
| **0** | 34143.236074 | 5.519894 | 3.488064 | 0.978780 | 1.986737 |
| **1** | 12216.216216 | 2.423423 | 0.950450 | 3.554054 | 6.878378 |
| **2** | 141040.000000 | 8.740000 | 0.600000 | 10.900000 | 1.080000 |

In [388…

```
fig, axes = plt.subplots(1, 5,  figsize=(16, 6))
fig.suptitle('Boxplot of numerical variables for each cluster', fontsize=20)
counter = 0
for ii in range(5):
    sns.boxplot(ax=axes[ii],y=subset_scaled_df[all_col[counter]],x=subset_scaled_df['HC
    counter = counter+1

fig.tight_layout(pad=2.0)
```




## Insights Hierarchical clustering

- **Cluster 0**:
  - This cluster contains customers with average `Avg_Credit_Limit` (between -0.5 and 0.5)
  - Also, average `Total_Credit_Cards` (between -0.2 and 0.7)
  - This cluster prefers to interact with the bank via bank visits, hence the `Total_visit_bank` is high.
  - This cluster does not prefers to interact with the bank via phone, nor online.
- **Cluster 1** :
  - This cluster contains customers with low `Avg_Credit_Limit` (less than -0.5)
  - Also, low `Total_Credit_Cards` (-0.8 or less)
  - This cluster prefers to interact with the bank via phone, hence the `Total_calls_made` is high.
  - This cluster does not prefers to interact with the bank via bank visits, nor online.
- **Cluster 2**:
  - This cluster contains customers with high `Avg_Credit_Limit` (between 2 and 3.5)
  - Also, high `Total_Credit_Cards` (between 1.5 and 3)
  - This cluster prefers to interact with the bank via online, hence the `Total_calls_made` is high.
  - This cluster does not prefers to interact with the bank via bank visits, nor phone.

In [ ]:

# Compare cluster K-means clusters and Hierarchical clusters - Cluster profiling

From the analysis above we can conclude that there si not much difference between K-Means Clustering and Hierarchical Clustering. Bot methods provided 3 clusters with similar charchteristics. And both methods have produced the same results.

Basically, in both methods, we have segmented the dataset in 3 clusters with the following profiles:

```
* Cluster 0 : is for Tier-1 accounts where customers have low
`Avg_Credit_limit` and less `Total_Credit_Cards`. This segment prefers to
interact with the bank by phone.

* Cluster 1 : is for Tier-2 accounts where customers have average
`Avg_Credit_limit` and average `Total_Credit_Cards`. This segment prefers
to interact with the bank by visits.

* Cluster 2 : is for Tier-3 accounts where customers have high
`Avg_Credit_limit` and high `Total_Credit_Cards`. This segment prefers to
interact with the bank online.
```

In [ ]:

## Actionable Insights & Recommendations

- Cluster 0 consists of Tier-1 account. The company can focus on offering more cards to this segment and by correlation, this segment can increate the `Avg_Credit_Limit` . Also, the bank can focus on providing better phone service support for this segment and cater to the products that this segment prefers.
- Cluster 1 consists of Tier-2 accounts. The company can provide better on-site support personel, since this segment prefers to physically visit the bank for their needs.
- Cluster 2 consists of Tier-3 accounts. The company can provide better on-line support, since this segment prefers to interact with the bank online.

In [ ]:

# End-of-File